PATENT ABSTRACTS OF JAPAN

(11)Publication number : 2002-099439
(43)Date of publication of application : 05.04.2002

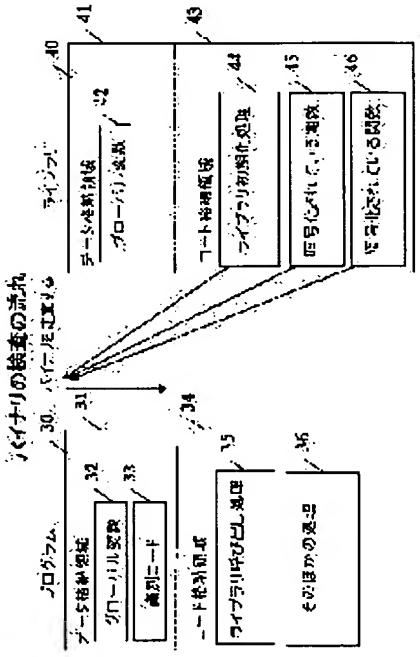
(51)Int.Cl. G06F 9/54
G06F 1/00

(21)Application number : 2000-287033
(22)Date of filing : 21.09.2000

(71)Applicant : NEC CORP
(72)Inventor : SATO TAKASHI

(54) LINK METHOD OF LIBRARY APPLICABLE TO COMPUTER SYSTEM AND RECORD MEDIUM RECORDED ITS PROGRAM

(57)Abstract:
PROBLEM TO BE SOLVED: To provide a link method of a library applicable to a computer system and a record medium recorded its program capable of calling a function by a link library only with an arbitrary executive program.
SOLUTION: In an executive program 30 to authorize a link to a library 40, a recognition code is placed, so that a key generated from the code encrypts a function that is in a high confidential among the library 40. When loading actually the library 40 from the program 30, since a control covers an initializing routine 44 in the library 40, at that time, an identification code 33 is searched with scanning a side of the program 30. Only when the code 33 in the program 30 is found, a calling of the function is made possible with expanding of functions 45 and 46 in the encrypted library.



(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号
特開2002－99439
（P2002－99439A）
(43)公開日　平成14年４月５日(2002.4.5)

(51)Int.Cl.'	識別記号	ＦＩ	ﾃｰﾏｺｰﾄﾞ(参考)
Ｇ０６Ｆ　9/54		Ｇ０６Ｆ　9/06	６４０Ｂ　５Ｂ０７６
1/00			６６０Ｇ

審査請求　有　請求項の数８　ＯＬ　（全 11 頁）

(21)出願番号　特願2000－287033(P2000－287033)

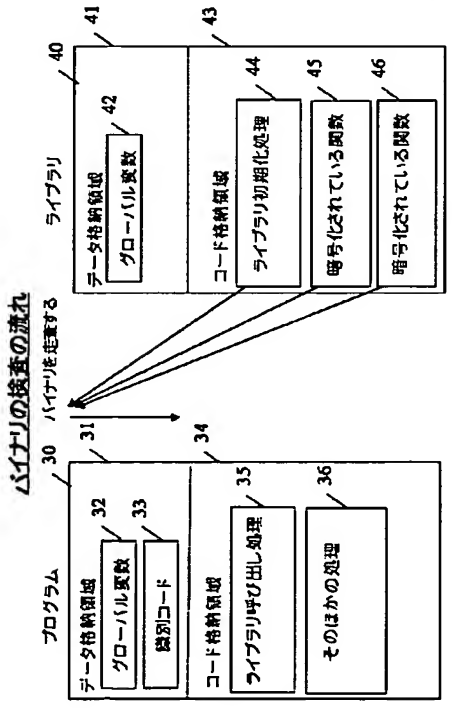
(22)出願日　平成12年９月21日(2000.9.21)

(71)出願人　000004237
日本電気株式会社
東京都港区芝五丁目７番１号
(72)発明者　佐藤　隆士
東京都港区芝五丁目７番１号　日本電気株式会社内
(74)代理人　100088328
弁理士　金田　暢之　（外２名）

Ｆターム(参考)　5B076 AB04 AB15 FA20

(54)【発明の名称】　コンピュータシステムに適用するライブラリのリンク方法及びそのプログラムを記録した記録媒体

(57)【要約】
【課題】　リンクライブラリが任意の実行プログラムによってのみ関数を呼び出すことができるようにする、コンピュータシステムに適用するライブラリのリンク方法及びそのプログラムを記録した記録媒体を提供する。
【解決手段】　ライブラリ４０にリンクを許可する実行プログラム３０中に、認識コードを置いておき、その認識コードから生成した鍵で、ライブラリ中の秘密性の高い関数を暗号化しておく。実際に実行プログラム３０から、ライブラリ４０をロードさせた場合は、ライブラリ４０中の初期化ルーチン４４に制御が渡るので、そのときに、実行プログラム３０側を走査して識別コード３３を探す。実行プログラム３０にある認識コードを見つけた場合にのみ、暗号化されているライブラリ中の関数４５，４６を展開して、関数の呼び出しが行えるようにする。



1

【特許請求の範囲】
【請求項１】　実行対象のプログラムと予め用意されたライブラリとのリンク処理をするためのリンク機能を有するコンピュータシステムに適用するライブラリのリンク方法であって、
実行プログラム側の処理であるライブラリの動的ロードによって該ライブラリ中の初期化ルーチンを呼ぶステップと、
該初期化ルーチンにより、実行プログラム側のバイナリイメージを順に走査して、識別コードを探すステップと、
該識別コードが正しいコードであった場合は、初期化処理を正常に完了するステップと、
前記識別コードが見つからなかった場合、または、不正な識別コードであった場合は、初期化処理を失敗とし、前記ライブラリをロードしないステップと、
実行プログラム側の処理である関数の呼び出しによって前記ライブラリ中の関数を呼ぶステップと、
該関数により、実行プログラム側のバイナリイメージを順に走査して、識別コードを探すステップと、
該識別コードが正しいコードであった場合は、該識別コードから暗号解除の鍵を得て、暗号化された関数を解除して関数を実行するステップと、
前記識別コードが見つからなかった場合、または、不正な識別コードであった場合は、ライブラリが不正な実行プログラムから呼ばれていると判断して、処理を中止するステップと、からなるライブラリのリンク方法。
【請求項２】　実行対象のプログラムと予め用意されたライブラリとのリンク処理をするためのリンク機能を有するコンピュータシステムに適用するライブラリのリンク方法であって、
実行プログラム側の処理であるライブラリの動的ロードによって該ライブラリ中の初期化ルーチンを呼ぶステップと、
該初期化ルーチンにより、実行プログラム側のバイナリイメージを順に走査して、識別コードを探すステップと、
該識別コードが正しいコードであった場合は、該識別コードから暗号解除の鍵を得て、暗号化された関数を解除して初期化処理を正常に完了するステップと、
前記識別コードが見つからなかった場合、または、不正な識別コードであった場合は、初期化処理を失敗とし、前記ライブラリをロードしないステップと、
実行プログラム側の処理である関数の呼び出しによって前記ライブラリ中の関数を呼ぶステップと、
該関数により、実行プログラム側のバイナリイメージを順に走査して、識別コードを探すステップと、
該識別コードが正しいコードであった場合は、該識別コードから暗号解除の鍵を得て、暗号化された関数を解除して関数を実行するステップと、

2

前記識別コードが見つからなかった場合、または、不正な識別コードであった場合は、ライブラリが不正な実行プログラムから呼ばれていると判断して、処理を中止するステップと、からなるライブラリのリンク方法。
【請求項３】　実行対象のプログラムと予め用意されたライブラリとのリンク処理をするためのリンク機能を有するコンピュータシステムに適用するライブラリのリンク方法であって、
実行プログラム側の処理である関数の呼び出しによって前記ライブラリ中の関数を呼ぶステップと、
該関数により、実行プログラム側のバイナリイメージを順に走査して、識別コードを探すステップと、
該識別コードが正しいコードであった場合は、該識別コードから暗号解除の鍵を得て、暗号化された関数を解除して関数を実行するステップと、
前記識別コードが見つからなかった場合、または、不正な識別コードであった場合は、ライブラリが不正な実行プログラムから呼ばれていると判断して、処理を中止するステップと、からなるライブラリのリンク方法。
【請求項４】　前記識別コードは、前記実行プログラムのデータ格納領域またはコード格納領域またはリソース格納領域に格納されている請求項１または２に記載のライブラリのリンク方法。
【請求項５】　実行対象のプログラムと予め用意されたライブラリとのリンク処理をするためのリンク機能を有するコンピュータシステムにより読み取り可能な記録媒体であって、
実行プログラム側の処理であるライブラリの動的ロードによって該ライブラリ中の初期化ルーチンを呼ぶステップと、
該初期化ルーチンにより、実行プログラム側のバイナリイメージを順に走査して、識別コードを探すステップと、
該識別コードが正しいコードであった場合は、初期化処理を正常に完了するステップと、
前記識別コードが見つからなかった場合、または、不正な識別コードであった場合は、初期化処理を失敗とし、前記ライブラリをロードしないステップと、
実行プログラム側の処理である関数の呼び出しによって前記ライブラリ中の関数を呼ぶステップと、
該関数により、実行プログラム側のバイナリイメージを順に走査して、識別コードを探すステップと、
該識別コードが正しいコードであった場合は、該識別コードから暗号解除の鍵を得て、暗号化された関数を解除して関数を実行するステップと、
前記識別コードが見つからなかった場合、または、不正な識別コードであった場合は、ライブラリが不正な実行プログラムから呼ばれていると判断して、処理を中止するステップとを、前記コンピュータシステムが実行するように設定されたプログラムを記録した記録媒体。

【請求項６】　実行対象のプログラムと予め用意されたライブラリとのリンク処理をするためのリンク機能を有するコンピュータシステムにより読み取り可能な記録媒体であって、
実行プログラム側の処理であるライブラリの動的ロードによって該ライブラリ中の初期化ルーチンを呼ぶステップと、
該初期化ルーチンにより、実行プログラム側のバイナリイメージを順に走査して、識別コードを探すステップと、
該識別コードが正しいコードであった場合は、該識別コードから暗号解除の鍵を得て、暗号化された関数を解除して初期化処理を正常に完了するステップと、
前記識別コードが見つからなかった場合、または、不正な識別コードであった場合は、初期化処理を失敗とし、前記ライブラリをロードしないステップと、
実行プログラム側の処理である関数の呼び出しによって前記ライブラリ中の関数を呼ぶステップと、
該関数により、実行プログラム側のバイナリイメージを順に走査して、識別コードを探すステップと、
該識別コードが正しいコードであった場合は、該識別コードから暗号解除の鍵を得て、暗号化された関数を解除して関数を実行するステップと、
前記識別コードが見つからなかった場合、または、不正な識別コードであった場合は、ライブラリが不正な実行プログラムから呼ばれていると判断して、処理を中止するステップとを、前記コンピュータシステムが実行するように設定されたプログラムを記録した記録媒体。
【請求項７】　実行対象のプログラムと予め用意されたライブラリとのリンク処理をするためのリンク機能を有するコンピュータシステムにより読み取り可能な記録媒体であって、
実行プログラム側の処理である関数の呼び出しによって前記ライブラリ中の関数を呼ぶステップと、
該関数により、実行プログラム側のバイナリイメージを順に走査して、識別コードを探すステップと、
該識別コードが正しいコードであった場合は、該識別コードから暗号解除の鍵を得て、暗号化された関数を解除して関数を実行するステップと、
前記識別コードが見つからなかった場合、または、不正な識別コードであった場合は、ライブラリが不正な実行プログラムから呼ばれていると判断して、処理を中止するステップとを、前記コンピュータシステムが実行するように設定されたプログラムを記録した記録媒体。
【請求項８】　前記識別コードは、前記実行プログラムのデータ格納領域またはコード格納領域またはリソース格納領域に格納されている請求項４から７のいずれか１項に記載の、前記コンピュータシステムが実行するように設定されたプログラムを記録した記録媒体。
【発明の詳細な説明】

【０００１】
【発明の属する技術分野】本発明は、コンピュータシステムに適用するライブラリのリンク方法及びそのプログラムを記録した記録媒体に関する。
【０００２】
【従来の技術】ソフトウェア開発では、同じプログラムを繰り返し作成する手間を省くために、似た働きのプログラムをまとめてライブラリとして用意する。ライブラリを使うプログラムは、コンパイル時にライブラリとまとめてコンパイルを行うことで、関数宣言だけでライブラリ中のプログラムコードを使用することができる。
【０００３】ライブラリは、生成される実行プログラムの容量が増大するという点と、ライブラリの機能を使うプログラム毎にコンパイルをしなくてはならないという欠点があった。これらを解決するために用いられるのが、動的なリンクライブラリである。
【０００４】動的なリンクライブラリは、プログラムコードの実行中に動的に呼び出される。プログラムの要求に応じてシステムはライブラリをメモリ上にロードする。プログラムはそのライブラリ中の関数を呼び出して、必要がなくなったら開放する。この方式により、どのプログラムコードでもコンパイルすることなくライブラリを使用することができ、しかも必要に応じてロードするので、実行プログラムのサイズを圧迫することはない。
【０００５】
【発明が解決しようとする課題】しかし、ある特定の実行ファイルのみライブラリの使用を許したいという場合には、この動的リンクライブラリの利点が、問題になる。どんなプログラムでも関数宣言さえできればライブラリ中のプログラムを使用することができるので、秘密性の高い関数や、ある特定のプログラムでないと正常に動作しないという関数の場合でも、他のプログラムから呼び出すことが可能になるからである。
【０００６】ライブラリが実行プログラムを選ぶシステムには、これまで、リンカによってプログラムを自動リンクさせる特開平06-083593号（1994-03-25）に開示された発明や、データ処理システムに設定されたセキュリティレベルに応じて、カーネルがサーバから動的リンクするライブラリを選択する特開平08-241207号（1996-09-17）に開示された発明などがあった。しかし前者は未解決の参照を探して解決するためのシステムであり、本発明の、特定プログラム以外とリンクをさせないという目的には用いることができない。後者はオペレーティングシステムを変更する必要があり、すでにあるシステムには対応できないという問題がある。
【０００７】また、ＣＡＤプログラムに対応する暗号化されたＣＡＤライブラリの内容をアクセスした場合に、正式なライセンスに基づくアクセス要求に対してのみ復号化したＣＡＤライブラリの情報を送出するＣＡＤライ

ブラリアクセス管理装置が特開2000-76313号（2000-03-14）に開示されている。そのＣＡＤライブラリアクセス管理装置の構成を図７に示す。 ＣＡＤライブラリアクセス管理装置１００は、ファイルシステム２００内に組み込まれているが、ファイルシステム２００において、リンクの許可を行う処理の大部分を行う。即ち、オペレーティングシステムなどに手を加える、もしくはそのようなオペレーティングシステムを用意する必要がある。また、図７からわかるように、ライブラリにおけるリンクの許可の有無を記載したライセンス情報６０やアクセス権管理対象パス定義情報５０などを別途用意する必要がある。

【０００８】本発明の目的は、以上の問題を解決する、リンクライブラリが任意の実行プログラムによってのみ関数を呼び出すことができるようにする、コンピュータシステムに適用するライブラリのリンク方法及びそのプログラムを記録した記録媒体を提供することにある。

【０００９】

【課題を解決するための手段】本発明のコンピュータシステムに適用するライブラリのリンク方法は、実行対象のプログラムと予め用意されたライブラリとのリンク処理をするためのリンク機能を有するコンピュータシステムに適用するライブラリのリンク方法であって、実行プログラム側の処理であるライブラリの動的ロードによってライブラリ中の初期化ルーチンを呼ぶステップと、初期化ルーチンにより、実行プログラム側のバイナリイメージを順に走査して、識別コードを探すステップと、識別コードが正しいコードであった場合は、初期化処理を正常に完了するステップと、識別コードが見つからなかった場合、または、不正な識別コードであった場合は、初期化処理を失敗とし、ライブラリをロードしないステップと、実行プログラム側の処理である関数の呼び出しによってライブラリ中の関数を呼ぶステップと、関数により、実行プログラム側のバイナリイメージを順に走査して、識別コードを探すステップと、識別コードが正しいコードであった場合は、識別コードから暗号解除の鍵を得て、暗号化された関数を解除して関数を実行するステップと、識別コードが見つからなかった場合、または、不正な識別コードであった場合は、ライブラリが不正な実行プログラムから呼ばれていると判断して、処理を中止するステップと、からなる。

【００１０】プログラムで使うライブラリには動的リンクライブラリというものがある。動的リンクライブラリは、プログラムの実行中に必要に応じてメモリ上にロードされ、ライブラリ中の関数を呼び出すことができる仕組である。動的リンクライブラリは一般に、実行プログラムの容量を節約するためや、よく使う関数を複数の実行プログラムで共有できるといった目的で使用される。

【００１１】本発明は、この動的リンクライブラリが、任意の実行プログラムからしかリンクできない機構を提供する。これにより、第三者に使用されたくない秘密性の高い関数を、任意の実行プログラムのみから動的にリンクできるライブラリとして利用することができる。

【００１２】ライブラリにリンクを許可する実行プログラム中に、認識コードを置いておき、その認識コードから生成した鍵で、ライブラリ中の秘密性の高い関数を暗号化しておく。

【００１３】実際に実行プログラムから、ライブラリをロードさせた場合は、ライブラリ中の初期化ルーチン制御が渡るので、そのときに、実行プログラム側を走査して識別コードを探す。実行プログラムにある認識コードを見つけた場合にのみ、暗号化されているライブラリ中の関数を展開して、関数の呼び出しが行えるようにする。

【００１４】これにより、識別コードのない第三者のプログラムでは、ライブラリの関数を使用することができなくなる。

【００１５】また、実行プログラム側の処理であるライブラリの動的ロードによってライブラリ中の初期化ルーチンを呼ぶステップと、初期化ルーチンにより、実行プログラム側のバイナリイメージを順に走査して、識別コードを探すステップと、識別コードが正しいコードであった場合は、識別コードから暗号解除の鍵を得て、暗号化された関数を解除して初期化処理を正常に完了するステップと、識別コードが見つからなかった場合、または、不正な識別コードであった場合は、初期化処理を失敗とし、ライブラリをロードしないステップと、実行プログラム側の処理である関数の呼び出しによってライブラリ中の関数を呼ぶステップと、関数により、実行プログラム側のバイナリイメージを順に走査して、識別コードを探すステップと、識別コードが正しいコードであった場合は、識別コードから暗号解除の鍵を得て、暗号化された関数を解除して関数を実行するステップと、識別コードが見つからなかった場合、または、不正な識別コードであった場合は、ライブラリが不正な実行プログラムから呼ばれていると判断して、処理を中止するステップと、からなってもよい。

【００１６】即ち、初期化ルーチンの時に識別コードから鍵を生成して、関数の暗号化を解除してもかまわない。関数が呼ばれたときにも暗号化を解除するので、二重の暗号化をかけることができる。

【００１７】また、実行プログラム側の処理である関数の呼び出しによってライブラリ中の関数を呼ぶステップと、関数により、実行プログラム側のバイナリイメージを順に走査して、識別コードを探すステップと、識別コードが正しいコードであった場合は、識別コードから暗号解除の鍵を得て、暗号化された関数を解除して関数を実行するステップと、識別コードが見つからなかった場合、または、不正な識別コードであった場合は、ライブラリが不正な実行プログラムから呼ばれていると判断し

て、処理を中止するステップと、からなってもよい。

【００１８】即ち、動的リンクライブラリでなくとも静的リンクライブラリでもかまわない。静的リンクライブラリを第三者に提供しなくてはならない場合に、そのライブラリに保護すべき内容があり、特定の実行ファイル以外にリンクさせたくない場合は、本発明の方法を使うことができる。静的リンクライブラリの場合は、初期化ルーチンが存在しないので、関数が呼ばれたときの方法だけを使うことになる。

【００１９】また、識別コードは、実行プログラムのデータ格納領域またはコード格納領域またはリソース格納領域に格納されていてもよい。

【００２０】即ち、実行ファイルに用意する認識コードはデータ格納領域でなくともかまわない。識別コードを複数用意して、格納する場所を分散させることで、解読されにくいシステムにすることができる。

【００２１】本発明のコンピュータシステムが実行するように設定されたプログラムを記録した記録媒体は、実行対象のプログラムと予め用意されたライブラリとのリンク処理をするためのリンク機能を有するコンピュータシステムにより読み取り可能な記録媒体であって、実行プログラム側の処理であるライブラリの動的ロードによってライブラリ中の初期化ルーチンを呼ぶステップと、初期化ルーチンにより、実行プログラム側のバイナリイメージを順に走査して、識別コードを探すステップと、識別コードが正しいコードであった場合は、初期化処理を正常に完了するステップと、識別コードが見つからなかった場合、または、不正な識別コードであった場合は、初期化処理を失敗とし、ライブラリをロードしないステップと、実行プログラム側の処理である関数の呼び出しによってライブラリ中の関数を呼ぶステップと、関数により、実行プログラム側のバイナリイメージを順に走査して、識別コードを探すステップと、識別コードが正しいコードであった場合は、識別コードから暗号解除の鍵を得て、暗号化された関数を解除して関数を実行するステップと、識別コードが見つからなかった場合、または、不正な識別コードであった場合は、ライブラリが不正な実行プログラムから呼ばれていると判断して、処理を中止するステップとを、コンピュータシステムが実行するように設定されている。

【００２２】また、実行プログラム側の処理であるライブラリの動的ロードによってライブラリ中の初期化ルーチンを呼ぶステップと、初期化ルーチンにより、実行プログラム側のバイナリイメージを順に走査して、識別コードを探すステップと、識別コードが正しいコードであった場合は、識別コードから暗号解除の鍵を得て、暗号化された関数を解除して初期化処理を正常に完了するステップと、識別コードが見つからなかった場合、または、不正な識別コードであった場合は、初期化処理を失敗とし、ライブラリをロードしないステップと、実行プログラム側の処理である関数の呼び出しによって前記ライブラリ中の関数を呼ぶステップと、関数により、実行プログラム側のバイナリイメージを順に走査して、識別コードを探すステップと、識別コードが正しいコードであった場合は、識別コードから暗号解除の鍵を得て、暗号化された関数を解除して関数を実行するステップと、識別コードが見つからなかった場合、または、不正な識別コードであった場合は、ライブラリが不正な実行プログラムから呼ばれていると判断して、処理を中止するステップとを、コンピュータシステムが実行するように設定されていてもよい。

【００２３】また、実行プログラム側の処理である関数の呼び出しによってライブラリ中の関数を呼ぶステップと、関数により、実行プログラム側のバイナリイメージを順に走査して、識別コードを探すステップと、識別コードが正しいコードであった場合は、識別コードから暗号解除の鍵を得て、暗号化された関数を解除して関数を実行するステップと、識別コードが見つからなかった場合、または、不正な識別コードであった場合は、ライブラリが不正な実行プログラムから呼ばれていると判断して、処理を中止するステップとを、コンピュータシステムが実行するように設定されていてもよい。

【００２４】

【発明の実施の形態】図１は、一般的な動的リンクライブラリの制御の流れを示した図である。プログラム１０とライブラリ２０から構成されている。実行プログラム１０は、ライブラリ２０中の関数をコールするプログラムである。プログラム１０は、システムにライブラリを動的にロードさせる、ライブラリの動的ロード１１の処理と、ライブラリ中の関数２２を使用するための関数の呼び出し１２の処理を含んでいる。

【００２５】ライブラリ２０は、動的リンクライブラリである。実行プログラムからコールされる関数２２、２３がある。

【００２６】ライブラリの動的ロード１１によって、制御はライブラリに渡り、初期化ルーチン２１の処理が行われる。これが終了すると、ふたたび制御はプログラム１０に戻る。関数呼び出し１２によって関数２２の呼び出しがおこなわれると、制御はライブラリ側になり、関数２２の処理を行う。これが終了するとプログラム側に制御が戻る。

【００２７】図２は、本発明を実施した実行プログラムとライブラリの例である。これらはプログラム３０とライブラリ４０から構成されている。

【００２８】プログラム３０は、データ格納領域３１とコード格納領域３４を含み、データ格納領域はグローバル変数３２と、識別コード３３を含む。コード格納領域は、ライブラリ呼び出し処理３５と、そのほかの処理３６を含んでいる。

【００２９】ライブラリ４０は、データ格納領域４１と

コード格納領域４３を含み、データ格納領域はグローバル変数４２を含む。コード格納領域はライブラリ初期化処理４４と暗号化された関数４５を含む。この暗号化を解除する鍵は、識別コード３３から生成できるようにしておく。

【００３０】本発明において、図２で構成されたシステムは、次のように動作する。

【００３１】図１のライブラリの動的ロード１１によってライブラリ２０中の初期化ルーチン２１が呼ばれる。図３のように、初期化ルーチン２１は、実行プログラム側のバイナリイメージ３０を順に走査して、識別コード３３を探す。これが正しいコードであった場合は、初期化処理を正常に完了する。識別コードが見つからなかったもしくは、不正な識別コードであった場合は、初期化処理は失敗し、ライブラリはロードされない。

【００３２】関数２２は図１の関数の呼び出し１２によって呼ばれると、プログラムのバイナリイメージ３０を検査して、識別コード３３を探す。この識別コードから暗号解除の鍵を得て、暗号化された関数４５を解除して関数２２に戻して関数を実行する。

【００３３】関数２３も同様の処理が行われる。

【００３４】識別コードが見つからない場合は、ライブラリが不正な実行プログラムから呼ばれていると判断して、処理を中止する。

【００３５】本発明でも図１の制御の流れは変わらない。ライブラリ側で行われる処理に追加して行う処理がある。本発明の全体の動作について、図４、図５のフローチャートを参照して、詳細に説明する。

【００３６】図４は、ライブラリの初期化処理が行う動作を示すフローチャートである。

・許可された実行プログラムの場合

図１のライブラリの動的ロード１１によって呼ばれたライブラリ初期化ルーチン２１では、まず自分を呼んだプログラムのバイナリコードを先頭から検査していく（ステップＡ１）。

【００３７】この検査によってプログラム中に置かれた識別コードを見つけたのならＡ３、そうでないのならＡ４へ分岐する（ステップＡ２）。

【００３８】得た識別コードが正しいのかどうかを検査する（ステップＡ３）。

【００３９】ライブラリを呼び出したプログラムは、許可されたプログラムではなかったので、初期化を失敗させる（ステップＡ４）。

【００４０】ライブラリを呼び出したプログラムは、許可されたプログラムだったので、初期化を成功させる（ステップＡ５）。

・識別コードを持たない実行プログラムの場合

自分を呼んだプログラムのバイナリコードを先頭から検査していく（ステップＡ１）。

【００４１】この検査によってプログラム中に識別コードが見つからないのでＡ４へ分岐する（ステップＡ２）。

【００４２】ライブラリを呼び出したプログラムは、許可されたプログラムではなかったので、初期化を失敗させる（ステップＡ４）。

【００４３】次に、図５は、プログラム実行中にライブラリの持つ関数をコールされたときの前処理の動作を示すフローチャートである。

・許可された実行プログラムの場合

図１の関数の呼び出し１２によって呼ばれたライブラリ中の関数は、まず自分を呼び出したプログラムのバイナリコードを先頭から検査していく（ステップＢ１）。

【００４４】この検査によってプログラム中に置かれた識別コードを見つけたのでＢ３へ分岐する（ステップＢ２）。

【００４５】識別コードは正しいことを検査したらＢ４へ分岐する（ステップＢ３）。

【００４６】識別コードから暗号解除の鍵を生成する（ステップＢ４）。

【００４７】その鍵で関数の一部にかけられた暗号化を解除する（ステップＢ５）。

【００４８】関数を呼び出したプログラムは、許可された実行プログラムだったので、関数の実行を行う（ステップＢ７）。

・識別コードを持たない実行プログラムの場合

まず自分を呼び出したプログラムのバイナリコードを先頭から検査していく（ステップＢ１）。

【００４９】この検査によって識別コードが見つからないのでＢ６へ分岐する（ステップＢ２）。

【００５０】関数を呼び出したプログラムは、許可された実行プログラムではなかったので、関数の実行ができない（ステップＢ６）。

【００５１】このように、ライブラリを呼び出すプログラム中に識別コードを置き、ライブラリ中の関数を、そのコードから生成した鍵で暗号化しておくことにより、識別コードのない実行プログラムが、ライブラリを呼び出して使用することを妨げることが可能になる。

【００５２】識別コードから鍵を生成する手順は、識別コードから論理演算などを複数回行って得た値を使用するのが望ましい。

【００５３】（発明の他の実施の形態）

・動的リンクライブラリでなくとも静的リンクライブラリでもかまわない。

【００５４】静的リンクライブラリを第三者に提供しなくてはならない場合に、そのライブラリに保護すべき内容があり、特定の実行ファイル以外にリンクさせたくない場合は、本発明の方法を使うことができる。静的リンクライブラリの場合は、初期化ルーチンが存在しないので、関数が呼ばれたときの方法だけを使うことになる。この場合、ライブラリ提供先に識別コードを知らせるの

は意味をなさないので、実行ファイル名などの既知の情報を識別コードにする必要がある。

・実行ファイルに用意する認識コードはデータ格納領域でなくともかまわない。

【００５５】図６の識別コードを分散して格納した例のように、プログラム・コードのパターンや、実行ファイルに格納された画像ファイルなどに認識コードを埋めてもかまわない。

【００５６】識別コードを複数用意して、格納する場所を分散させることで、解読されにくいシステムにすることができる。この場合、識別コードの種類は鍵の種類になる。関数１つにつき１種類の鍵を割り当てると、ひとつの識別コードが知られても他の暗号化した関数に影響を及ぼさないライブラリができる。また、すべての関数で複数の鍵による暗号化を行うようにすれば、単体のセキュリティ強度の高いライブラリになる。

・初期化ルーチンの時に識別コードから鍵を生成して、関数の暗号化を解除してもかまわない。

【００５７】関数が呼ばれたときにも暗号化を解除するので、二重の暗号化をかけることができる。この場合は鍵は共通で、暗号化アルゴリズムが異なるものを用意する必要がある。

【００５８】

【発明の効果】以上説明したように、本発明の効果は、任意の実行プログラム以外からは、その関数をコールすることのできない動的リンクライブラリの作成を可能にすることにある。

【００５９】その理由は、動的リンクライブラリが初期化ルーチンを通過したとき、更に、実行プログラムから関数をコールされたときに、実行プログラム側に埋め込まれている識別コードを走査して、動的リンクを許可したプログラムかどうかを検査することにある。

【００６０】そして識別コードがライブラリの暗号化を解除する鍵になっていることにある。

【００６１】また、動的リンクライブラリでなくとも静的リンクライブラリでもかまわない。

【００６２】静的リンクライブラリを第三者に提供しなくてはならない場合に、そのライブラリに保護すべき内容があり、特定の実行ファイル以外にリンクさせたくない場合は、本発明の方法を使うことができる。

【００６３】また、実行ファイルに用意する認識コードはデータ格納領域でなくともかまわない。識別コードを複数用意して、格納する場所を分散させることで、解読されにくいシステムにすることができる。

【００６４】また、初期化ルーチンの時に識別コードから鍵を生成して、関数の暗号化を解除してもかまわない。関数が呼ばれたときにも暗号化を解除するので、二重の暗号化をかけることができる。

【図面の簡単な説明】

【図１】動的ライブラリ使用時の一般的な制御の図である。

【図２】本発明を適用したライブラリと実行ファイルのバイナリイメージの図である。

【図３】本発明を適用したライブラリ初期化ルーチンのバイナリ検査の図である。

【図４】初期化ルーチンのバイナリ検査の処理のフローチャートである。

【図５】関数が実行される前処理のフローチャートである。

【図６】識別コードを分散して格納する場合の構成図である。

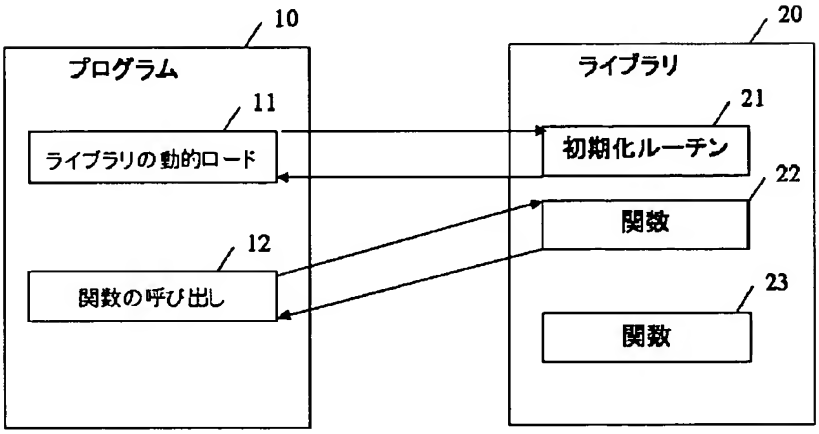
【図７】従来の技術のＣＡＤライブラリアクセス管理装置の構成を示す図である。

【符号の説明】

１０　プログラム
１１　システムにライブラリをロードさせる処理
１２　ライブラリの関数を呼び出す処理
２０　ライブラリ
２１　初期化ルーチン
２２～２３　呼び出される関数
３０　プログラム
３１，４１　データ格納領域
３２，４２　グローバル変数
３３　識別コード
３４，４３　コード格納領域
３５　システムにライブラリをロードさせる処理
３６　そのほかの処理
３７　識別コードを格納したそのほかの処理
３８　リソース格納領域
３９　識別コードを格納した画像データ
４０　ライブラリ
４４　ライブラリ初期化処理
４５，４６　暗号化されている関数
５０　アクセス権管理対象パス定義情報
６０　ライセンス情報
１００　ＣＡＤライブラリアクセス管理装置
１０２　アクセス先パス確認処理部
１０４　アクセス権確認処理部
１０６　ライブラリ復号化処理部
１１０　ファイルアクセス処理部
１２０　ファイルアクセス要求元
１３０　暗号化されたライブラリ
１４０　アクセス管理されていないファイル
２００　ファイルシステム

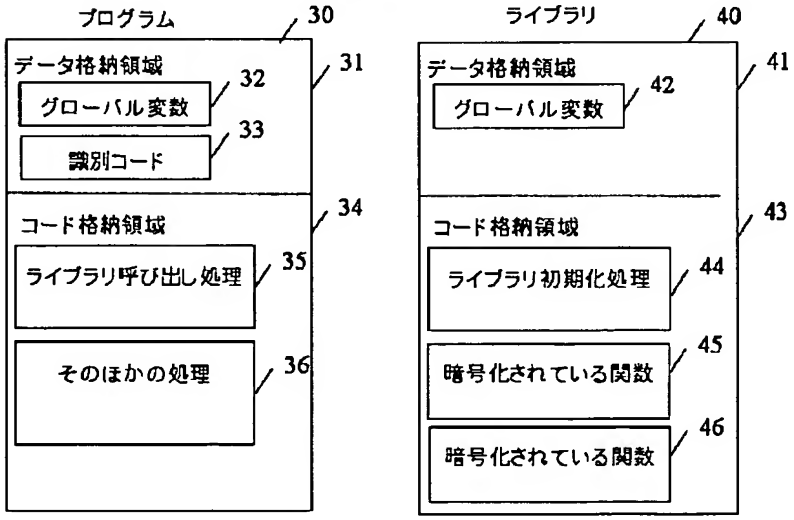
【図１】

動的ライブラリ使用時の制御の流れ

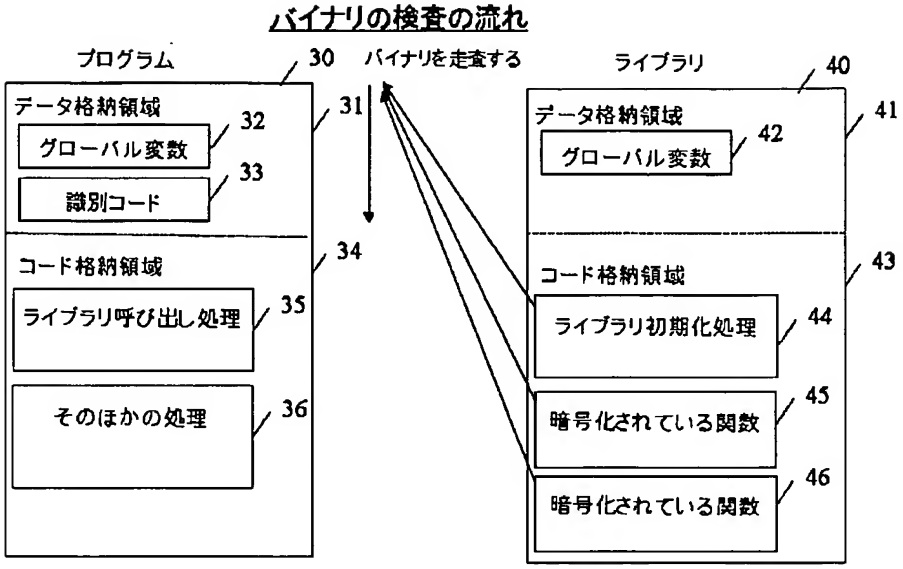


【図２】

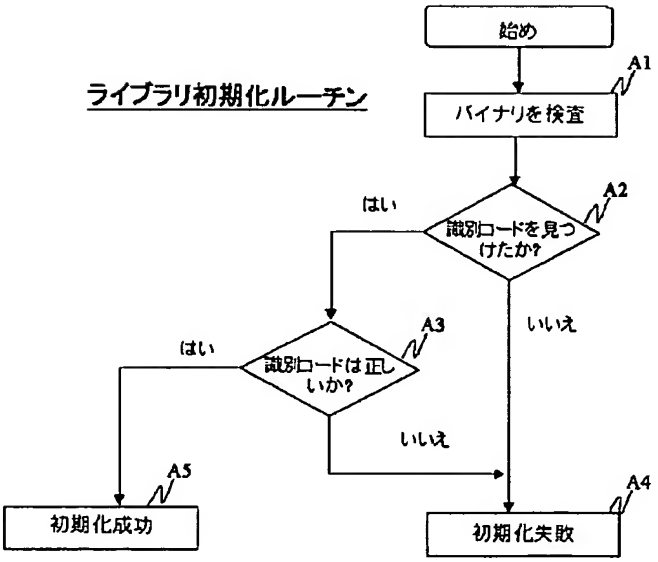
本発明の構成



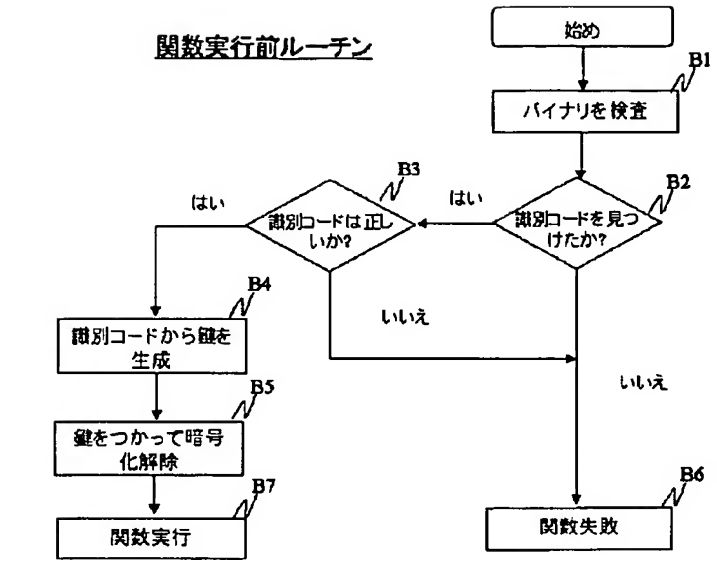
【図３】


バイナリの検査の流れ
プログラム
30
バイナリを走査する
ライブラリ
40
データ格納領域
31
32
グローバル変数
33
識別コード
41
データ格納領域
42
グローバル変数
コード格納領域
34
43
コード格納領域
ライブラリ呼び出し処理
35
ライブラリ初期化処理
44
そのほかの処理
36
暗号化されている関数
45
暗号化されている関数
46

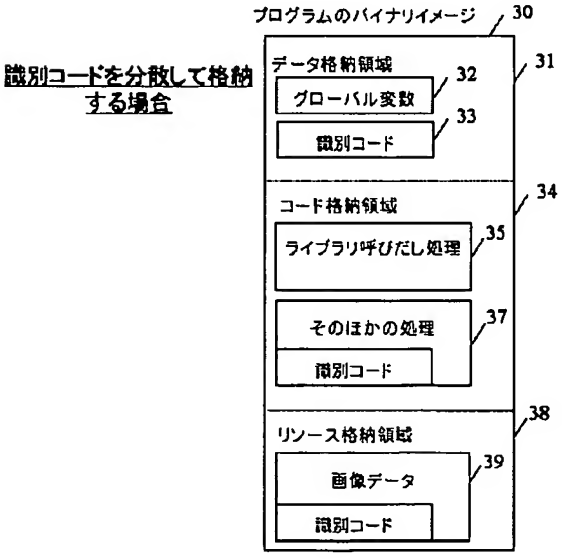
【図４】


ライブラリ初期化ルーチン
始め
A1
バイナリを検査
A2
はい
識別コードを見つけたか?
いいえ
A3
はい
識別コードは正しいか?
いいえ
A5
初期化成功
A4
初期化失敗

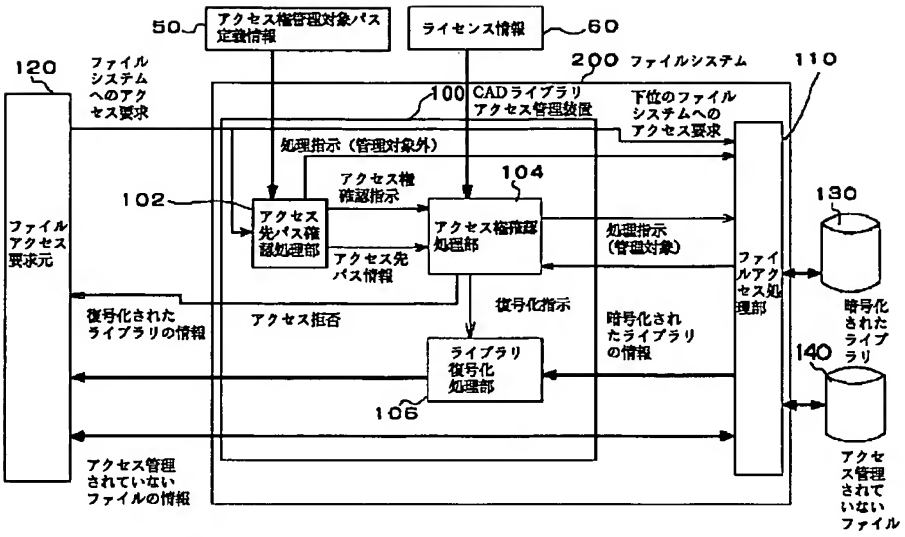
【図５】


関数実行前ルーチン
始め
B1
バイナリを検査
B2
識別コードを見つけたか?
はい
B3
識別コードは正しいか?
はい
B4
識別コードから鍵を生成
いいえ
B5
鍵をつかって暗号化解除
B7
関数実行
いいえ
B6
関数失敗

【図６】


識別コードを分散して格納する場合
プログラムのバイナリイメージ
30
データ格納領域
31
グローバル変数
32
識別コード
33
コード格納領域
34
ライブラリ呼びだし処理
35
そのほかの処理
37
識別コード
リソース格納領域
38
画像データ
39
識別コード

【図７】


50
アクセス権管理対象パス定義情報
ライセンス情報
60
120
ファイルシステムへのアクセス要求
200 ファイルシステム
110
100 CADライブラリアクセス管理装置
下位のファイルシステムへのアクセス要求
処理指示（管理対象外）
アクセス権確認指示
104
102
アクセス先パス確認処理部
アクセス権確認処理部
処理指示（管理対象）
ファイルアクセス処理部
130
ファイルアクセス要求元
アクセス先パス情報
復号化指示
復号化されたライブラリの情報
アクセス拒否
暗号化されたライブラリの情報
暗号化されたライブラリ
140
ライブラリ復号化処理部
106
アクセス管理されていないファイルの情報
アクセス管理されていないファイル